EMBEDDING A SECURITY SUPPORT PROVIDER INTERFACE IN A COMMUNICATION CLASS LIBRARY

CROSS REFERENCE TO CO-PENDING APPLICATIONS

U.S. Patent Application No. ______, filed _____, and entitled, "Cool ICE data Wizard"; U.S. Patent Application No. ______, filed _____, and entitled, "Cool ICE Column Profiling"; U.S. Patent Application No. ______, filed _____, and entitled, "Cool ICE OLEDB Consumer Interface"; and U.S. Patent Application No. ______, filed _____, and entitled, "Cool ICE State Management" are commonly assigned co-pending applications.

BACKGROUND OF THE INVENTION

1. <u>Field of the Invention</u>: The present invention generally relates to data base management systems and more particularly relates to enhancements for improving the efficiency of secure access to data base management systems.

2. <u>Description of the prior art</u>: Data base management systems are well known in the data processing art. Such commercial systems have been in general use for more than 20 years. One of the most successful data base management systems is available from Unisys Corporation and is called the Classic MAPPER® data base management system. The Classic MAPPER system can be reviewed using the Classic MAPPER User's Guide which may be obtained from Unisys Corporation.

The Classic MAPPER system, which runs on proprietary hardware also available from

Unisys Corporation and on an industry compatible personal computer under a Windows Server operating system, provides a way for clients to partition data bases into structures called filing cabinets and drawers, as a way to offer a more tangible format. The BIS (Business Information System) data base manager utilizes various predefined high-level instructions whereby the data base user may manipulate the data base to generate human-readable data presentations called "reports". The user is permitted to prepare lists of the various predefined high-level instructions into data base manager programs called "BIS Runs":. Thus, users of the Classic MAPPER system may create, modify, and add to a given data base and also generate periodic and aperiodic reports using various BIS Runs.

Within these highly complex network and multi-legacy environments, standardization of security profiling becomes a particular problem. It is known in the prior art to utilize Security Support Provider Interface (SSPI) available from Microsoft. Unfortunately in the current environment, this means that the client and server must ordinarily first establish their connection through a communications library (e.g., sockets). The system then requires a multi-step SSPI handshake to authenticate both client and server. Following the mutual identification, the SSPI functions must be individually called to protect the security and integrity of each message.

SUMMARY OF THE INVENTION

The present invention overcomes the disadvantages of the prior art by providing a method of and apparatus for improving the efficiency of client server communication within a secure environment. The preferred mode of the present invention embeds use of a commercial security facility, such as Microsoft Security Support Provider Interface (SSPI) within a communications class library, so that the communications library supports peer authentication, client impersonation, and message signature and encryption. It is this embedded SSPI which provides the security features for the client/server relationship. The client and server can then authenticate each other, and sign or encrypt messages between them.

In the preferred approach, the use of SSPI is hidden in a generic communications object. Clients and servers do not directly call any SSPI functions. The connection and authentication appear to occur in a single step, and the applications send and receive messages without concern about encryption and decryption. Applications may want to encrypt some messages but not others. The communications library provides methods to turn signing and encryption on and off.

The communications class, CDACSComm, has subclasses CDACSCommClient and CDACSCommServer for the client and server to use, respectively. It has the CDACSSecurity object embedded into it. The client application creates and initializes a CDACSCommClient application. Initialization includes the information needed to identify the server, and provides choices, with default values, for authentication and message protection.

The client then calls the Open method, with a simple message that the server can use to route the connection. (In DACS, we have three different server applications that can receive the connection from a single listener application.) The security sublibrary provided by this invention adds the authentication and encryption selections to the initial Open message.

For its part, the server application creates a CDACSCommServer object and initializes it with a token that represents a tentatively accepted client connection and its choices for authentication and message protection. At this point, the security library takes over, making sure that client and server agree on their authentication choices. It performs the steps needed to carry out the authentication, calling SSPI functions and sending messages between client and server as needed. Both the client and server applications receive a simple status back indicating whether the connection is fully established.

Once the connection is established and authenticated, the client and server applications send and receive messages as though they were plain, unencrypted text. The communications library signs or encrypts sent messages, and verifies or decrypts received messages automatically. The applications need not be aware that they are using a security sublibrary.

BRIEF DESCRIPTION OF THE DRAWINGS

Other objects of the present invention and many of the attendant advantages of the present invention will be readily appreciated as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, in which like reference numerals designate like parts throughout the figures thereof and wherein:

Fig. 1 is a pictographic view of the hardware of the preferred embodiment;

Fig. 2 is a pictorial diagram of the @SPI command process flow;

Fig. 3, consisting of **Fig. 3A**, **Fig. 3B**, and **Fig. 3C**, is a main class diagram showing embedding of the preferred SSPI functions;

Fig. 4 is a detailed flow diagram showing an authorizing connection; and

Fig. 5 is a table showing the description of the message utilized in Fig. 5.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention is described in accordance with several preferred embodiments which are to be viewed as illustrative without being limiting. These several preferred embodiments are based upon Unisys Intel-based hardware and Microsoft Windows operating systems, the Classic MAPPER data base management system, and the BIS/ Cool ICE software components, all available from Unisys Corporation.

Fig. 1 is a pictorial diagram of hardware suite 10 of the preferred embodiment of the present invention. The client interfaces with the system via terminal 12. Preferably, terminal 12 is an industry compatible, personalized computer having a current version of the Windows operating system and suitable web browser, all being readily available commercial products. Terminal 12 communicates over world wide web access 16 using standardized HTML protocol, via Server 14.

The BIS/Cool ICE system is resident in Enterprise Server 20 and accompanying storage subsystem 22, which is coupled to Server 14 via WAN (Wide Area Network) 18. In the preferred mode, Server 14 is owned and operated by the enterprise owning and controlling the proprietary legacy data base management system. Server 14 functions as the Internet access provider for terminal 12 wherein world wide web access 16 is typically a dial-up telephone line. This would ordinarily be the case if the shown client were an employee of the enterprise. On the other hand, web server 14 may be a remote server site on the Internet if the shown client has a different Internet access provider. This would ordinarily occur if the shown client were a customer or guest.

In addition to being coupled to WAN 18, Enterprise Server 20, containing the BIS/Cool ICE system, is coupled to departmental server 24 having departmental server storage facility 26. Additional departmental servers (not shown) may be sinilarly coupled. The enterprise data and enterprise data base management service functionality typically resides within enterprise server 20, departmental server 24, and any other departmental servers (not shown). Normal operation in accordance with the prior art would provide access to this data and data base management functionality.

In the preferred mode of the present invention, access to this data and data base management functionality is also provided to users (e.g., terminal 12) coupled to Intranet 18. As explained below in more detail, server 14 provides this access utilizing the BIS/Cool ICE system.

Fig. 2 is a functional diagram showing the major components of the @SPI (stored procedure interface) command process flow. This command is a part of the MRI (BIS Relational Interface) set of commands and combines many of the attributes of the previously existing @FCH (relational aggregate fetch) and @SQL (standard query language) commands. However, it is specifically targeted to executing stored procedures.

Command set 28 represents the commands defined for processing by MRI. In addition to @SPI , @FCH, and @SQL, @LGN (log on), MRI recognizes @LGF (log off), @DDI (data definition information), @RAM (relational aggregate modify), @TRC (trace relational syntax), @MQL (submit SQL syntax to a BIS data base) as the remaining commands. DAC/BIS core Engine 30 provides the basic logic for decode and execution of these commands. MRI 34 has relational access to data via the data base management formats shown to external data bases 40. In addition, MRI 34 can call upon remote MRI 38 to make similar relational access of remote data bases 42.

BIS core engine 30 executes commands utilizing meta-data library 32 and BIS repository 36. Meta-data library 32 contains information about the data within the data base(s). BIS repository 36 is utilized to store command language script and state information for use during command execution.

The @SPI command has the following basic format:

@SPI, c, d, lab, db, edsp?, action, wrap, vert 'sp-syntax', vpar1.....,vparN, typ1,....typN.

Fields c and d refer to the cabinet and drawer, respectively, which hold the result. The lab field contains a label to go to if the status in the vstat variable specifies other than normal completion.

The required db field provides the data base name. The edsp? field specifies what is to be done with the result if an error occurs during execution.

The sub-field labeled action defines what action is to be performed. The options include execution, return of procedures lists, etc. The wrap sub-field indicates whether to truncate or wrap the results. The vert sub-field defines the format of the results. The name of the stored procedure is placed into the sp-syntax field. The vpar provides for up to 78 variables that correspond to stored procedure parameters. Finally, the typ field defines the type of each stored procedure parameter.

Fig. 3 containing **Fig. 3A**, **Fig. 3B**, and **Fig. 3C**, provides a detailed class diagram for the preferred mode of the present invention. Turning to Fig. 3C, it can be seen that the communication is initiated by the listener via the CommListener object 506, the client via the CommClient ojbect 508, or the server via the CommServer object 510. These three objects are refinements of Comm object 494, as shown in Fig. 3B. CommState object 496 maintains the state of the particular communication. Control is provided by object 500 which also contains objects 498 and 502. The communications header object is element 504. Error handling functions are found at object 492 (see also Fig. 3A).

Fig. 3A shows the objects associated with the specific security functions via object 484. Error handling functions are found at object 486. Object 488 provides authorization type. Message protection types are defined by object 490.

Fig. 4 is a detailed schematic view of the process for authorizing a connection. Element 512 represents the communication client and element 514 represents the communication server. First message 516 is the initial request from client to server in the format shown as element 532. Message 518 provides the initial response from the server in the format of element 534. Message 520 occurs within the client to obtain a context token. Message 522, in the format of element 536, is sent to the server to provide the context token. The server internalizes the context token via message 524. The status of the communication (e.g., accepted, impersonate client, etc.) is sent from server to client via message 526 in the format of element 538. Element 542 indicates that the client is impersonated as necessary. If required, element 540 shows repeat of steps 3-6.

Message 528 is a second message to the server from the client. It is in the format of element 544. Message 530 provides the server response in the format of element 546.

Fig. 5 is a listing and description of all of the messages and corresponding operations shown within Fig. 4.

Having thus described the preferred embodiments of the present invention, those of skill in the art will be readily able to adapt the teachings found herein to yet other embodiments within the scope of the claims hereto attached.

WE CLAIM: